PENERAPAN NEURAL NETWORK TENTANG METODE BACKPROPAGATION PADA PENGENALAN POLA HURUF

Asep Sholahuddin, MT

Jurusan Matematika Universitas Padjadjaran
Jl. Raya Jatinangor Km21. Telp. (022) 7794696 Bandung
E-mail : asol@math.unpad.ac.id dan asol@melsa.net.id

Abstrak

Neural network adalah salah satu cabang dari Artificial Intelligent. Salah satu metode dalam neural network adalah metode backpropagation. Banyak aplikasi dari neural network diantaranya : untuk prediksi, pengenalan pola, identifikasi dan simulasi. Dalam paper ini, neural network dengan menggunakan metode backpropagation telah diaplikasikan dengan sukses untuk pengenalan pola huruf abjad. Untuk mencobanya dibuat software, software tersebut ditraining untuk mengenali pola huruf setelah itu dicoba untuk mengenali huruf yang dimasukan, ternyata software tersebut mengenalinya bahkan dicoba huruf-huruf tersebut diberi noise, tapi dalam batas-batas tertentu tetap masih mengenali huruf tersebut. Dari metode tersebut bisa dikembangkan lebih jauh lagi, misalnya pengenalan pola wajah, pengenalan pola tanda tangan dan lain-lain

Kata kunci: neural network, backpropagation, noise.

1. Pendahuluan

Bila seseorang membaca huruf, misalnya huruf A maka orang tersebut akan mudah sekali membacanya tetapi bagaimana kalau huruf tersebut ada beberapa bagian yang terhapus, orang tersebut akan mencoba memperkirakan huruf tersebut dan kemungkinan bisa menebaknya dengan baik. Kenapa manusia bisa seperti itu? jawabannya karena manusia mempunyai pengenalan pola yang sangat baik. Bagaimana halnya dengan komputer?. Maka pada paper ini saya mencoba komputer untuk bisa mengenali pola.dengan menggunakan Jaringan Syaraf Tiruan (JST).

Aplikasi JST pada saat ini telah merambah banyak bidang. Misalnya, pada pesawat terbang: sistem kontrol pesawat, deteksi kesalahan, simulasi lintasan terbang, pada otomotif: sistem pemandu otomatis, pada pertahanan: deteksi musuh, elektronika: prediksi urutan kode, pada hiburan: animasi, efek khusus, pada keuangan: prediksi harga terbaru, program penjualan portofolio, optimalisasi poduk, pada manufaktur: kontrol proses manufaktur, analisis dan desain produk, pada kedokteran: analisis sel kanker, analisis EEG dan ECG, pada minyak dan gas : bidang eksplorasi, pada suara : pengenalan suara, kompresi suara, klasifikasi vokal, pada saham : sistem penasihat perdagangan saham dan lain-lain.

2. Metode Jalar Balik (*backpropagation*)

Metode yang digunakan dalam kesempatan ini adalah metode jalar balik dimana metode ini sangat populer meskipun bukan yang terbaik. Metode ini tidak mempunyai hubungan *feedback* sehingga galat dijalar-balik selama latihan lalu diperoleh Galat Kuadrat Rerata Terkecil. Galat dalam output menentukan ukuran galat output lapisan hidden yang digunakan untuk menentukan bobot antara input dan lapisan hidden. Penentuan bobot antara pasangan-pasangan lapisan dan mengkalkulasi ulang output merupakan proses iteratif yang dilakukan sampai galat mencapai toleransi tertentu. Bobot penentuan parameter laju belajar dan momentum yang sesuai bisa digunakan untuk memperbaiki hasil dari JST.

Pasangan vektor input dan output dipilih untuk melatih JST untuk pertama kalinya. Setelah latihan selesai, maka bobot ditentukan dan JST dapat digunakan untuk menemukan output sebagai

input baru. Sejumlah neuron dalam lapisan input menentukan dimensi input, dan sejumlah neuron dalam lapisan output menentukan dimensi lapisan output, kemudian JST dapat membuat pemetaan ruang dimensi-k menjadi ruang dimensi-m. Tentu saja, pemetaan tersebut tergantung pada bagaimana pasangan pola atau vektor digunakan. Sehingga setelah dilatih, JST akan memberikan vektor input baru.

Tidak mudah untuk menentukan berapa banyak neuron yang diperlukan. Oleh sebab itu kita membagi tiga bidang, satu untuk neuron input, satu untuk unsur pemroses hidden, dan satu untuk neuron output, sehingga menjadi hubungan jalar maju.

JST jalar-balik mengalami latihan tanpa guru, dengan sejumlah pasangan pola berhingga yang terdiri dari pola input dan sebuah pola output target atau yang diinginkan. Pola input muncul pada lapisan input lalu ke lapisan berikutnya yaitu pada lapisan hidden. Output neuron hidden layer diperoleh dengan menggunakan bias, dan juga fungsi ambang dengan aktivasi yang ditentukan dengan bobot dan input. Output lapisan hidden tersebut menjadi input ke neuron output, yang juga diproses menggunakan sebuah fungsi bias dan fungsi ambang dengan aktivasinya untuk menentukan output akhir dari JST.

Pola terhitung dan pola input lalu dibandingkan sehingga galat untuk masing-masing komponen pola dapat ditentukan, sementara penentuan bobot koneksi antara lapisan hidden dan lapisan output dapat dihitung. Sebuah perhitungan serupa (masih berdasarkan pada galat pada output) dibuat untuk bobot koneksi antara input dan lapisan hidden. Proses kemudian diulang sebanyak yang diperlukan sampai galat dalam batas toleransi yang diinginkan.

Algoritma jalar balik adalah sbb:

-Inisialisasi Bobot

Menentukan bobot dan ambang simpul dengan sedikit acak.

-Menghitung aktivasi

Menentukan tingkat aktivasi input.

Tingkat aktivasi O_j sebuah hidden dan satuan output ditentukan oleh:

$$O_j = F (\sum W_{ji}O_i - \theta_j) \quad (1)$$

Dimana W_{ji} adalah bobot dari input O_i , θ_j adalah ambang simpul dan F adalah fungsi sigmoid :

$$F(a) = 1/ (1+ e^{-a}) \quad (2)$$

-Melatih bobot

1. Mulai pada output terus pada lapisan hidden secara rekursif. Pemasangan bobot oleh:

   $$W_{ji}(t+1)=W_{ji}(t)+ \Delta W_{ji} \quad (3)$$

   Dimana $W_{ji}(t)$ adalah bobot dari unit i ke unit j pada waktu t (atau iterasi ke t) dan ΔW_{ji} adalah penentuan bobot.

2. Perubahan bobot dihitung dengan :

   $$\Delta W_{ji} = \eta \delta_j O_i$$

   dimana η adalah laju belajar ($0<\eta<1$ misal 0,3) dan δ_j adalah gradien galat pada unit j. Konvergensi kadang-kadang lebih cepat dengan menambahkan bentuk momentum:

   $$W_{ji}(t+1)=W_{ji}(t)+\eta\delta_j O_i + \alpha[W_{ji}(t)- W_{ji}(t-1)] \quad \text{dimana } 0<\alpha<1 \quad (4)$$

3. Gradien galat diberikan dengan:

   -Untuk unit output:

   $$\delta_j=O_j(1-O_j)(T_j-O_j) \quad (5)$$

   dimana T_j adalah aktivasi output (target) yang diinginkan dan O_j adalah aktivasi output aktual pada unit output j.

   -Untuk unit hidden:

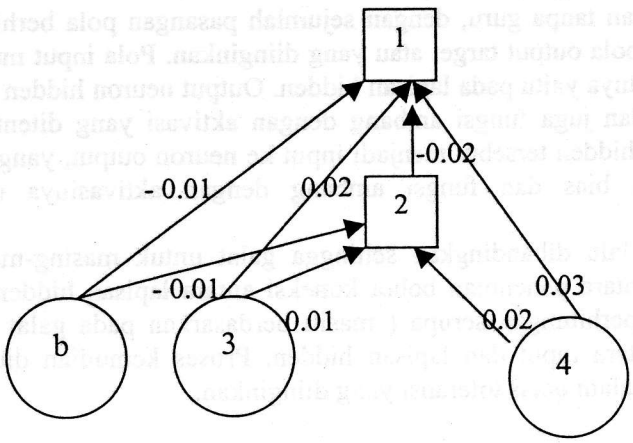
   $$\delta_j=O_j(1-O_j)\sum\delta_k W_{kj} \quad (6)$$

   δ_k adalah gradien galat pada unit k dimana merupakan titik hubung dari unit hidden j.

4. Mengulangi iterasi sampai konvergen untuk kriteria galat tertentu. Iterasi termasuk menampilkan contoh, menghitung aktivasi, dan memodifikasi bobot.

Nama Jalar-balik berasal dari kenyataan dimana galat diturunkan dari jalar galat terbalik yang diasosiasikan dengan unit output (sebagaimana dihitung berdasarkan persamaan diatas) ketika nilai target untuk unit hidden tidak diberikan. Pada JST jalar-balik, fungsi aktivasi yang dipilih adalah fungsi sigmoid, yang menentukan nilai output ke dalam range 0 dan 1.

Contoh metode backpropagasi yang diterapkan pada masalah ekslusive-or



Input	output
(1,1)	0
(1,0)	1
(0,1)	1
(0,0)	0

Gambar 1. penyelesaian eklusive-or dengan menggunakan metode backpropagasi

Kita inisialisasi bobot :

W_{13}=0.02, W_{14}=0.03, W_{12}= - 0.02, W_{23}=0.01, W_{24}=0.02, W_{1b}= - 0.01, W_{2b}= - 0.01
Kalkulasi aktifasi untuk input (1,1) yang menghasilkan output (0)

$O_3=O_4=1$

$O_2=1/[1+e^{-(1 \times 0.01 + 1 \times 0.02 - 1 \times 0.01)}]=0.505$

$O_1=1/[1+e^{-(0.505 \times -0.02 + 1 \times 0.03 - 1 \times 0.01)}]=0.508$

Traning bobot dengan η=0.3

$\delta_1 = 0.508(1 - 0.508)(0 - 0.508)= -0.127$

$\Delta W_{13} = 0.3 \times (-0.127) \times 1 = - 0.038$

$\delta_2 = 0.505(1 - 0.505)(-0.127 \times - 0.02)= 0.0006$

$\Delta W_{23} = 0.3 \times 0.0006 \times 1 = 0.0002$

iterasi terus dilakukan dan berakhir pada error lebih kecil dari 0.01 didapatkan bobot-bobot sbb:

W_{13}=4.98, W_{14}=4.98, W_{12}= - 11.30, W_{23}=5.62, W_{24}=5.62, W_{1b}= - 2.16, W_{2b}= - 8.83

(untuk programnya dapat dilihat di lampiran1)

3. Hasil dan Pembahasan Mengenali Huruf

Pada kesempatan ini JST didesain dan dilatih untuk mengenali 26 huruf alpabet. Masing-masing huruf diwakili oleh nilai Boolean kisi 5x7.

Pertama kita mendefinisikan 35 vektor input dan vektor target dimana masing-masing vektor target adalah 26 vektor dengan 1 pada posisi huruf yang mewakilinya dan 0 untuk yang lainnya. Sebagai contoh, huruf A diwakili oleh 1 pada elemen pertama (untuk alasan kemudahan, karena A adalah huruf pertama alpabet) dan 0 pada elemen 2 sampai 26.

Tabel 1 JST yang dilatih untuk mengenali huruf alpabet

No	Huruf	Input	Output
1	A	0 0 1 0 0 0 1 0 1 0 0 1 0 1 0 1 0 0 0 1 1 1 1 1 1 1 0 0 0 1 1 0 0 0 1	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2	B	1 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 1 1 1 0	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
⋮	⋮	⋮	⋮
26	Z	1 1 1 1 1 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1 1 1 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

JST terdiri dari 35 vektor input dan 26 vektor output. Masing-masing vektor output mewakili sebuah huruf. JST yang benar mampu merespon salah satu huruf yang ada sementara nilai output lainnya nol. JST yang baik juga seharusnya dapat mengatasi noise dimana JST seharusnya membuat kesalahan sekecil mungkin ketika mengklasifikasi vektor bernoise rata-rata 0 dan deviasi standar 0,2 atau kurang.

Tabel 2 JST ketika diuji untuk mengenali 3 huruf ideal maupun 3 huruf bernoise

No	Uji JST dgn data ini (ideal & bernoise)	Kesimpulan JST	berarti
1	0 0 1 0 0 0 1 0 1 0 0 1 0 1 0 1 0 0 0 1 1 1 1 1 1 1 0 0 0 1 1 0 0 0 1 tanpa noise	0.9 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0	A
2	0.0 0.0 0.2 0.0 0.0 0.0 0.7 0.0 0.6 0.0 0.0 0.0 0.6 0.0 0.8 0.0 0.3 0.0 0.0 0.0 0.2 0.2 0.6 0.0 0.2 0.6 0.3 0.0 0.0 0.0 0.1 0.0 0.0 0.0 0.0 0.1 diberi noise random	0.8 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0	A
3	0 1 1 1 0 1 0 0 0 1 1 0 0 0 0 1 0 0 0 0 1 0 0 1 1 1 0 0 0 1 0 1 1 1 0 tanpa noise	0.0 0.0 0.0 0.0 0.0 0.0 0.9 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0	G
4	0.6 0.0 0.0 0.0 0.5 0.9 0.0 0.0 0.0 0.5 0.8 0.0 0.0 0.0 0.1 0.9 0.1 0.7 0.6 0.1 0.0 0.0 0.0 0.0 0.2 0.9 0.0 0.0 0.0 0.3 0.2 0.0 0.0 0.0 0.7 diberi noise random	0.0 0.0 0.0 0.0 0.0 0.0 0.6 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0	G
5	1 1 1 1 0 1 0 0 0 1 1 0 0 0 1 1 1 1 1 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0 1 tanpa noise	0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.9 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0	R
6	0.4 0.2 0.0 0.2 0.0 0.9 0.0 0.0 0.0 0.6 0.8 0.0 0.0 0.0 0.2 0.0 0.3 0.6 0.0 0.0 0.9 0.0 0.7 0.0 0.0 0.2 0.0 0.0 0.6 0.0 0.8 0.0 0.0 0.0 0.8 diberi noise	0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.2 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0	R

JST memerlukan 35 input dan 26 neuron pada lapisan output untuk mengidentifikasi huruf. JST terdiri dari dua lapisan. Fungsi transfer log-sigmoid dipilih karena jangkauan outputnya 0 sampai 1 yang cocok untuk latihan nilai Boolean output.
Lapisan hidden terdiri dari 10 neuron. Jumlah ini dipilih berdasarkan tebakan dan pengalaman. Jika JST bermasalah dalam latihan maka neuron dapat ditambahkan pada lapisan ini.
JST dilatih untuk satu output pada vektor output yang tepat dan mengisi vektor output yang lain dengan 0. bagaimanapun vektor input bernoise mungkin hasilnya tidak sempurna 0 dan 1.

Untuk menghasilkan JST yang dapat mengatasi vektor input bernoise cara terbaik yaitu melatih JST baik ideal maupun vektor bernoise. Oleh sebab itu JST pertama kali dilatih pada vektor ideal sampai mempunyai SSE kecil. Kemudian JST akan dilatih pada 10 set ideal dan vektor bernoise. JST dilatih pada alpabet bebas noise rangkap dua dimana pada saat yang sama dilatih pada vektor bernoise. Alpabet bebas noise rangkap dua digunakan untuk memelihara kemampuan JST untuk mengklasifikasi vektor input ideal. JST akan kembali dilatih pada vektor ideal. Hal ini untuk meyakinkan bahwa JST akan merespon secara sempurna ketika muncul huruf ideal. Seluruh latihan dikerjakan dengan menggunakan jalar balik.

Untuk latihan tanpa noise, JST pada awalnya dilatih tanpa noise untuk maksimum 5000 epoch atau sampai SSE mencapai 0,1. Sementara untuk bernoise, kita latih dengan ideal rangkap dua dan vektor bernoise rangkap dua pada alpabet. Vektor target terdiri dari vektor rangkap 4 pada target. Vektor noise mempunyai noise rata-rata 0,1 dan 0,2 yang ditambahkan pada sistem. Hal ini akan membuat neuron belajar bagaimana mengidentifikasi huruf bernoise lebih tepat, tetapi tetap merespon vektor ideal dengan baik.

Untuk melatih bernoise masimum maka jumlah epoch dikurangi sampai 3000 dan galat goal dinaikkan sampai 0,6. Hal ini dikarenakan jumlah vektor semakin banyak.

Setelah JST dilatih dengan noise maka masuk akal untuk melatihnya tanpa noise. Hal ini untuk lebih meyakinkan bahwa vektor input ideal selalu diklasifikasi secara benar. Karenanya JST dilatih lagi dengan pola identik sebagaimana yang pertama kalinya.

Kehandalan sistem pengenalan pola JST diukur dengan menguji JST dengan ratusan vektor input dengan jumlah noise yang bervariasi.

JST tidak membuat kesalahan untuk vektor dengan noise rata-rata 0,00 atau 0,05. ketika noise rata-rata 0,2 ditambahkan untuk vektor maka kedua JST mulai membuat kesalahan.

Jika akurasi lebih tinggi diperlukan maka JST dapat dilatih ulang dengan lebih banyak neuron pada lapisan hiddennya. Juga resolusi vektor input dapat dibaikan katakanlah 10x14. akhirnya JST dapat dilatih pada vektor input dengan jumlah noise lebih besar jika kehandalan lebih tinggi diperlukan untuk tingkat noise yang lebih tinggi (untuk programnya dapat lihat lampiran2).

4. Kesimpulan

Telah ditunjukkan bahwa JST mampu mengenali dengan baik jenis-jenis huruf dari A sampai Z, bahkan ketika diberi noise sekalipun. Hal ini memberikan kemungkinan yang besar untuk perkembangan pengenalan pola yang lebih jauh lagi.

5. Daftar Pustaka

[1] Blum, A, *Neural Networks in C++*, New York:John Wiley & Son, 1992
[2] Chester, *Neural Networks: A Tutorial*, New Jersey: Prentice Hall, Inc, 1994
[3] Demoth, Howard, *Neural Networks Toolbox*, The Mathworks, Inc, Massachusetts, 1995
[4] Eberhart, Russel C, Dobbins, *Neural Networks PC Tools: A Practical Guide*, San Diego:Academic Press, Inc, 1990
[5] Fu Li Min, *Neural Networks in Computer Intelligence*, Singapore:McGraw-Hill, Inc, 1994
[6] Haykin, Simon, *Neural Networks: A Comprehensive Foundation*, New Yorks: Macmillan, Inc, 1994
[7] Rao, Valluru, *C++ Neural Networks and Fuzzy Logic*, New York:MIS Press, 1993
[8] *The Student Edition of Matlab version 4*, The Mathwork, Inc, Massachusetts, 1995

Lampiran1. Listing program EXOR dengan software Matlab

```
input=[1 1;1 0;0 1;0 0]';
output=[0 1 1 0];

[R,Q] = size(input);
```

```
[S2,Q] = size(output);

hidden = 10;
net = newff(minmax(input),[hidden S2],{'logsig' 'logsig'},'traingdx');

net.performFcn = 'sse';
net.trainParam.goal = 0.001;
net.trainParam.epochs = 5000;
net.trainParam.mc = 0.95;

P = input;
T = output;

[net,tr] = train(net,P,T);


contoh=[1 1]';

    A = sim(net,contoh);
```

Lampiran2. Listing program mengenali pola huruf dengan software Matlab

```
[alphabet,targets] = jenishuruf;
[R,Q] = size(alphabet);
[S2,Q] = size(targets);

%pause % tekan sembarang tombol...


S1 = 10;
net = newff(minmax(alphabet),[S1 S2],{'logsig' 'logsig'},'traingdx');
net.LW{2,1} = net.LW{2,1}*0.01;
net.b{2} = net.b{2}*0.01;


net.performFcn = 'sse';
net.trainParam.goal = 0.01;
net.trainParam.show = 20;
net.trainParam.epochs = 5000;
net.trainParam.mc = 0.95;


P = alphabet;
T = targets;

[net,tr] = train(net,P,T);



contoh =  [1 0 0 0 1 ...
           1 0 0 0 1 ...
           1 0 0 0 1 ...
           1 1 1 1 1 ...
           1 0 0 0 1 ...
           1 0 0 0 1 ...
           1 0 0 0 1 ]';
```

```
    A = sim(net,contoh);

function [alfabet,target] = prprob()

hurufA =    [0 0 1 0 0 ...
             0 1 0 1 0 ...
             0 1 0 1 0 ...
             1 0 0 0 1 ...
             1 1 1 1 1 ...
             1 0 0 0 1 ...
             1 0 0 0 1 ]';

hurufB =    [1 1 1 1 0 ...
             1 0 0 0 1 ...
             1 0 0 0 1 ...
             1 1 1 1 0 ...
             1 0 0 0 1 ...
             1 0 0 0 1 ...
             1 1 1 1 0 ]';

hurufC =    [0 1 1 1 0 ...
             1 0 0 0 1 ...
             1 0 0 0 0 ...
             1 0 0 0 0 ...
             1 0 0 0 0 ...
             1 0 0 0 1 ...
             0 1 1 1 0 ]';

hurufD  =   [1 1 1 1 0 ...
             1 0 0 0 1 ...
             1 0 0 0 1 ...
             1 0 0 0 1 ...
             1 0 0 0 1 ...
             1 0 0 0 1 ...
             1 1 1 1 0 ]';
.....

hurufZ =    [1 1 1 1 1 ...
             0 0 0 0 1 ...
             0 0 0 1 0 ...
             0 0 1 0 0 ...
             0 1 0 0 0 ...
             1 0 0 0 0 ...
             1 1 1 1 1 ]';

alfabet = [hurufA,hurufB,hurufC,hurufD,hurufE,hurufF,hurufG,hurufH,...
           hurufI,hurufJ,hurufK,hurufL,hurufM,hurufN,hurufO,hurufP,...
           hurufQ,hurufR,hurufS,hurufT,hurufU,hurufV,hurufW,hurufX,...
           hurufY,hurufZ];

target = eye(26);
```